

CS320- Test1 Review

- $A \setminus B = x$ exists in A but not B
- $P(A)$, P of A is the set of subsets of A
- **If set A has n elements in it then there are 2^n possible subset combinations**
- Every subset has a binary string representation
- There are no duplicates in a set and two sets are equal if they have the same elements
- An ordered pair is a set
- A function from $A \rightarrow B$ is defined as f is a subset of $A \times B$
- Range = the set of values actually assigned vs. Target = all possible outcomes
- Injective Function = every element in the domain maps to exactly one range in the target (1-1)
- The inverse of a function means that $f(A) \rightarrow A$
- A function must be injective in order for an inverse function to exist
- The empty set \emptyset is a subset of every set
- The cardinality of A is less than the cardinality of B if there exists injection from A to B
- For the cardinality of A to equal the cardinality of B need injection from A to B and injection from B to A
- Infinite Set = contains a proper subset of the same cardinality
- Aleph null has the same cardinality as the set of natural numbers, and it is the first smallest infinite cardinality (but is countable).
- Any set with cardinality less than or equal to aleph null is said to be countable and finite
- The set of subsets of \mathbb{N} is uncountable and infinite (injection exists from $\mathbb{N} \rightarrow P(\mathbb{N})$ but not from $P(\mathbb{N}) \rightarrow \mathbb{N}$)
- Aleph null programs exist, countable many programs exist (a program is a sequence of letters and there are countably many finite sequences)
- A problem is a set and there are uncountably many problems
- Countable programs for uncountably many problems
- Goedel number needs to contain a sequence of sequential primes- can't be an odd number
- If a word has K letters in it, then there are $k + 1$ possible splits
- Total function is defined everywhere vs. partial function, only a subset of the domain is defined
- Regular Expression = a string over the alphabet $\{a, b, c, \text{lamda}, \text{fi}, (,), U, \text{concatenation}, *\}$ (# of elements + 7)
- The class of regular languages over Σ is defined as: empty set, $\{\text{lamda}\}$, singleton sets

- Grammar = generates a language: $G = \{V, \sigma, P, S\}$
- The language $L(G)$ derived by G is the set of terminal strings that are derivable from S in finite # of steps. Makes a language with aleph null strings in it: infinite but countable
- Every regular expression is context free
- Regular grammar is context free if it follows the form: $A \rightarrow a$, $A \rightarrow \lambda$, $A \rightarrow aB$
Ex. $S \rightarrow \lambda \mid aS \mid bS \quad (aUb)^*$
- A regular language can have regular or not regular CFG. If you see a grammar that is not regular, you don't know anything about the language it generate
- NEVER USE C AS A VARIABLE NAME
- Automata = recognizes languages: $M = \{Q, \sigma, \delta, q_0, F\}$
- $L(M)$ is the set of strings accepted by M
- $\Delta(q, \lambda)$ is the terminal configuration, its accepting if q is a final state, otherwise its rejecting
- DFA- δ is total, NFA- δ is partial
- Deterministic Finite Automata are a special case of non-deterministic finite automata where δ is total instead of partial
- NFA = can have more than one way to go, can have λ transitions, may be missing some transitions. Accepts a string that potentially leads to acceptance in one way but not in all ways

1. $G_1 + G_2$

- $U = \{S \rightarrow S_1 \mid S_2\}$
- Concatenation = $\{S \rightarrow S_1S_2\}$
- $*$ = $\{S \rightarrow \lambda \mid SS \mid S_1\}$

2. Regular Expression \rightarrow Grammar

- Base case: $S \rightarrow \lambda$, $S \rightarrow a$, fi- no rule
- If the regular expression has operators then use algorithm 1

3. Regular Expression \rightarrow NFA (can't have any incoming or outgoing arcs)

- Base case: λ , singleton, fi – draw their automata
- If there are operators, combine union, concat, and $*$ automata combos

4. Non-Deterministic Finite Automata \rightarrow Deterministic Finite Automata

- Make a transition state grid and include a column for $C(x)$

- b. Create a new transition state grid with $C(x)$ as the new states (f_i is a state)
 - c. Draw the new Deterministic Finite Automata
 - d. The new final state is any state that contains the old final state
5. Regular Grammar \rightarrow Automata (needs to have a single final state with no out degrees, and no λ arcs except to the final state)
 - a. $q_0 = S$
 - b. $F = \{Z\}$ //need one more state that isn't a variable of the grammar
 - c. Convert to proper form
 - d. Combine the arcs and states (ex. $A \rightarrow aB$, $B \rightarrow \lambda$)
6. Automata \rightarrow Regex (no in or out degrees, need on final state)
 - a. Use GEG (generalized expression graph) to eliminate nodes
 - b. Create state transition grids using the regular expressions as the transitions

Problem Solving

- For regexes with a specified range (ex. more than 3 but less than 6), use λ as one of your options once you've reached the min quota
- For a compliment, do the unacceptable case and append to it

Counting

- Set of all **strings over σ^*** = aleph null
- Set of all non-empty strings over σ^* = aleph null
- Set of all **regular expressions over σ** = aleph null
- Set of all **context free grammar over σ** = aleph null
- Set of all **context free languages over σ** = aleph null
- Set of all **finite languages over σ** = aleph null
- Set of all **finite subsets over σ^*** = aleph null
- Set of all **infinite subsets over σ^*** = greater than aleph null
- Set of all **non-empty languages over σ** = greater than aleph null
- Set of **languages over σ** whose cardinality is greater than 3 = greater than aleph null
- Set of all **regular languages over σ** = aleph null
- Set of all languages over σ that contain exactly three strings = aleph null

- Set of all **strings over sigma** whose length is greater than 3 = aleph null
- L = number of choices in the regex times each other
- **L^* = aleph null**
- **Σ^* = aleph null**
- Complement of L (in Σ^*) = aleph null
- **$P(L^*)$ = greater than aleph null**
- **$P(\Sigma)$, the set of subsets of sigma = 2 raised to the number of elements in sigma**
- Set of all subsets of Σ^* (**$P(\Sigma^*)$**) = greater than aleph null
- Set of all non-empty subsets of Σ^* = greater than aleph null
- Set of all non-empty subsets over sigma = answer - 1
- Set of **total functions over sigma** = number of elements in the set raised to elements in the sigma
- Set of **total functions over Σ^*** = greater than aleph null
- **$(\Lambda)(f)(a) = 0$**
- **$\Lambda \cup f \cup a = 1$**
- **$(\Lambda^*)(f^*)(a^*) = \text{aleph null}$**
- **$F^* = 1$ (since it has lamda)**